Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika
Kolokvijum: Prvi, decembar 2023.
Datum: 03. 12. 2023.

Prvi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____%

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Raspoređivanje procesa

U nekom sistemu koristi se raspoređivanje procesa po MFQS na sledeći način. Postoje tri reda spremnih procesa, red višeg prioriteta HP, red srednjeg prioriteta MP i red nižeg prioriteta LP. Kada proces dolazi u red spremnih iz stanja suspenzije ili prvi put kada je tek pokrenut (parametar `wasBlocked=true` operacije `Scheduler::put`), stavlja se u red HP. Kada se proces u red spremnih vraća zbog isteka vremenskog odsečka, vraća se u isti red u kom je bio prethodni put, ali se to ponavlja najviše `AGE_THR` puta: ako je proces već bio u istom redu u istom naletu izvršavanja `AGE_THR` puta, onda se smešta u prvi niži red (osim za red LP, u kom ostaje do kraja naleta izvšravanja). Raspoređivanje u svakom redu je *Round Robin*.

Klasa `Scheduler`, čiji je interfejs dat dole, realizuje opisani raspoređivač spremnih procesa. Implementirati u potpunosti ovu klasu tako da i operacija dodavanja novog spremnog procesa `put()` i operacija uzimanja spremnog procesa koji je na redu za izvršavanje `get()` budu ograničene po vremenu izvršavanja vremenom koje ne zavisi od broja spremnih procesa (kompleksnost $O(1)$). U slučaju da nema drugih spremnih procesa, operacija `get()` vraća *null*. Polje `next` strukture `PCB` služi za ulančavanje u liste. Navesti precizno potrebna proširenja strukture `PCB`.

```
class Scheduler {
public:
  Scheduler ();
  PCB* get ();
  void put (PCB*, bool wasBlocked);
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Napraviti monitor koji svojim operacijama *startRead*, *stopRead*, *startWrite* i *stopWrite* obezbeđuje protokol „više čitalaca – jedan pisac“ (engl. *multiple readers – single writer*), ali tako da prednost daje piscima (i time potencijalno izgladnjuje čitaoce): čitaoci se ne puštaju da počnu čitanje ako ima pisca koji čeka. Koristiti klasične uslovne promenljive proširene samo operacijom *signalAll* koja deblokira sve procese koji čekaju na uslovnoj promenljivoj i jednog po jednog pušta da nastavi svoju započetu operaciju monitora pre ostalih procesa.

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Napisati kod servera koji omogućava prijavljivanje korisnika. Korisnik ima samo korisničko ime. Korisnik se registruje na serveru prilikom prve prijave. Nakon uspešnog prijavljivanja može da koristi server. U jednom trenutku korisnik može da ima najviše tri aktivne prijave. Ako korisnik prekorači taj broj, prijava treba da se odloži sve dok se broj aktivnih prijava ne smanji. Kada uspešno završi prijavljivanje može da od servera traži željene usluge (obradu usluga nije potrebno implementirati, samo staviti komentar na mesto na kom bi taj kod bio). Nakon obrade svih usluga korisnik se odjavljuje sa servera. Server treba da ima mogućnost obrade zahteva od više klijenata uporedo. Server osluškuje na portu 5555.

Rešenje: